 **GAYATRI VIDYA PARISHAD COLLEGE FOR**

**DEGREE AND PG COURSES (AUTONOMOUS)**

**(AFFILIATED TO ANDHRA UNIVERSITY|ACCRREDITED BY NACC WITH ‘A’ GRADE)**

**(MBA AND UG ENGINEERING B.TECH(CE,CSE,ECE AND ME) PROGRAMS ARE ACCREDITED BY NBA)**

**VISAKHAPATNAM - 530045.**

**PROJECT ON:**

**PUMPKIN SEEDS PREDICTION USING MACHINE LEARNING**

**SUBMITTED BY**

VEERAMALLA SAI HARSHA VARDHAN (2023-2415132)

VULLI HEMA LATHA(2023-2415107)

KONGARANI ANUSHA (2023-2415179)

NANUBALA VENKATESH(2023-2415143)

SHAIK KAREEMULLASHA(2023-2415220)

B.SC COMPUTER SCIENCE[HONS]

Under the Esteemed Guidance of

**DECLARATION**

I hereby declare that the Project work entitled “PUMPKIN SEEDS PREDICTION USING MACHINE LEARNING” is being submitted to Gayatri Vidya Parishad College for Degree and PG Courses (Autonomous) in partial fulfillment for the award of B.Sc Computer Science (Honours).

This work was originally designed and executed by me under the guidance of Dr./Mr./Ms. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, from the Department of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, and is not a duplication of work done by anyone else.

I hold full responsibility for the originality of the work incorporated into this report. The content presented in this dissertation has not been submitted for the award of any other degree. All technical details and concepts provided herein are purely relevant to the scope and objectives of the project and align with theoretical and practical aspects of machine learning-based predictive modeling.

NAME:

VEERAMALLA SAI HARSHA VARDHAN (2023-2415132)

VULLI HEMA LATHA(2023-2415107)

KONGARANI ANUSHA (2023-2415179)

NANUBALA VENKATESH(2023-2415143)

SHAIK KAREEMULLASHA(2023-2415220)

**CERTIFICATE**

This is to certify that the project entitled “PUMPKIN SEEDS PREDICTION USING MACHINE LEARNING”, that is being submitted by VEERAMALLA SAI HARSHA VARDHAN (2023-2415132), VULLI HEMA LATHA(2023-2415107), KONGARANI ANUSHA (2023-2415179), NANUBALA VENKATESH(2023-2415143), SHAIK KAREEMULLASHA(2023-2415220) in partial fulfilment for the award of B.Sc Computer Science (Honours) degree during the academic year \_\_\_\_\_\_\_\_\_, at Gayatri Vidya Parishad College for Degree and PG Courses (Autonomous), is a record of bonafide work carried out by him under my guidance and supervision.The results embodied in this work have not been submitted to any other University or Institute for the award of any degree or diploma.

**SIGNATURE OF GUIDE SIGNATURE OF HOD**

**ACKNOWLEDGEMENT**

I would like to express my heartfelt gratitude to all those who guided and supported me during the successful completion of my project titled **“PUMPKIN SEEDS PREDICTION USING MACHINE LEARNING.”** I am especially thankful to my project guide, **Dr. --------------------**, Head of the Department of --------------------, for their constant support, expert guidance, and encouragement throughout the duration of this work. I also take this opportunity to thank our respected Principal, **Dr. --------------------**, for providing the necessary facilities and a supportive environment to carry out the project effectively. My sincere thanks to the **faculty members and technical staff** of the Department of --------------------, whose assistance and valuable suggestions greatly enriched this project.

I extend my gratitude to the **management** of **-------------------- College** for creating a conducive learning atmosphere and for their continuous motivation.

I am deeply thankful to my **family and friends** for their unconditional support, patience, and moral encouragement during all phases of the project.This project has been a significant learning experience, giving me insight into real-world machine learning applications in engineering diagnostics.

| **TITLE** | **PAGE NO** |
| --- | --- |
| **ABSTRACT** | 7 |
| **1. INTRODUCTION** | 8 |
| 1.1 Aim and Objective | 9 |
| 1.2 Scope of the Project | 10 |
| 1.3 Problem Statement | 11-13 |
| **2. SYSTEM ANALYSIS AND DESIGN** | 14-16 |
| 2.1 Existing System | 14 |
| 2.2 Proposed System | 14 |
| 2.3 Functional Requirements | 15 |
| 2.4 Non-Functional Requirements | 16 |
| **3. METHODOLOGY & IMPLEMENTATION** | 17-19 |
| 3.1 Dataset Collection | 17 |
| 3.2 Feature Selection & Preprocessing | 17 |
| 3.3 Machine Learning Algorithms Used | 17 |
| 3.4 Model Training and Evaluation | 18 |
| 3.5 Streamlit Application for Prediction | 19 |
| **4. SYSTEM DESIGN** | 20-26 |
| 4.1 System Architecture | 21 |
| 4.2 Data Flow Diagram | 22 |
| 4.3 UML Diagrams | 23 |
| 4.3.1 Use Case Diagram | 24 |
| 4.3.2 Class Diagram | 25 |
| 4.3.3 Activity Diagram | 26 |
| **5. IMPLEMENTATION** | 27-53 |
| 5.1 Sample Code | 27-46 |
| 5.2 Screenshots | 47-53 |
| 5.3 Output Results | 53 |
| **6. TESTING** | 54 |
| 6.1 Unit Testing | 54 |
| 6.2 Integration Testing | 54 |
| 6.3 System Testing | 54 |
| **7. CONCLUSION** | 55-56 |
| **8. REFERENCES** | 57 |

**ABSTRACT**

Seed classification is a crucial task in agricultural technology, especially for quality control and automated sorting systems. Accurate and early identification of seed types can optimize processing, improve yield, and ensure consistency in agricultural production. In this project, we present a machine learning-based solution to classify pumpkin seed varieties using morphological features extracted from digital images.

The dataset used in this study consists of various geometric and shape descriptors such as Area, Perimeter, Major and Minor Axis Lengths, Eccentricity, Solidity, Roundness, and Aspect Ratio. These features were employed as input variables, while the seed class served as the target label for supervised classification.

We implemented and evaluated several classification algorithms including Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). Performance was assessed using metrics such as Accuracy, Precision, Recall, and F1-score.

Additionally, the project features an interactive Streamlit-based dashboard that allows users to explore seed features, visualize model predictions, and compare classifier performance. This approach shows promise in automating seed classification tasks and can be extended to other crop species and real-time agricultural applications.

**1. INTRODUCTION**

Accurate identification of agricultural products such as seeds plays a vital role in ensuring quality, boosting productivity, and supporting modern farming practices. Misclassification can lead to inefficiencies in processing and reduce crop quality. One effective solution is to use machine learning techniques to classify seeds based on their physical characteristics.

This project, “**Pumpkin Seed Classification Using Machine Learning**”, aims to automatically identify the type of pumpkin seed using various morphological features extracted from seed images. The features include attributes like **Area, Perimeter, Major and Minor Axis Lengths, Eccentricity, Solidity, Roundness, Aspect Ratio, and more**. These features help distinguish between different pumpkin seed varieties.

In this project:

* We use a dataset containing labeled measurements of different pumpkin seed types.
* We apply classification algorithms such as **Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN)**.
* We evaluate which model provides the highest accuracy in classifying seed types.
* We provide an interface for users to input new feature values and get predictions on the seed class.

This system is valuable for agricultural processing units, researchers, and seed distributors by automating the classification process, reducing manual errors, and increasing operational efficiency.

**1.1 AIM AND OBJECTIVE**

The primary objective of this project is to develop an efficient and accurate machine learning system for classifying pumpkin seed varieties based on their morphological characteristics. The system leverages advanced classification algorithms to automate the identification process, thereby enhancing quality control and streamlining operations in agricultural settings.

**Specific objectives include:**

* To analyze the **physical and geometric features of pumpkin seeds** using a structured dataset.
* To implement and compare multiple machine learning algorithms for classification, including **Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN)**.
* To identify the most effective algorithm in terms of **accuracy and reliability** for seed classification.
* To build an interactive platform that allows users to **input seed feature values and receive real-time predictions**.
* To support automation in **seed sorting and reduce dependency** on manual inspection.

**Benefits of the proposed system:**

* Increases **operational efficiency** in agricultural production lines.
* Provides a **scalable solution** for real-time seed classification.
* Enables **data-driven decision-making** for seed distributors, researchers, and agronomists.

**1.2 SCOPE OF THE PROJECT**

This project is designed to work with a dataset of pumpkin seed features and focuses on building a machine learning model to classify different types of pumpkin seeds. The system supports agricultural analysis and quality control by predicting the seed variety based on measurable input features such as:

* Area
* Perimeter
* Major Axis Length
* Minor Axis Length
* Eccentricity
* Convex Area
* EquivDiameter
* Extent
* Solidity
* Roundness
* Aspect Ratio
* Compactness

The scope of this project includes:

* **Data collection and preprocessing:** Cleaning and preparing the pumpkin seed dataset for training, including encoding categorical labels and scaling numerical features.
* **Model training and testing:** Applying classification algorithms such as Logistic Regression, K-Nearest Neighbors, Support Vector Machine, Random Forest, and XGBoost.
* **Performance comparison:** Evaluating models based on classification accuracy to determine the most effective approach.
* **User interface development:** Building an interactive and beginner-friendly web application using Streamlit that allows users to input seed measurements and get a predicted class instantly.

**Problem Statement**

In agricultural industries, the classification of seeds is a critical task that impacts product quality, sorting efficiency, and downstream processing. Traditionally, seed classification is performed manually or through basic rule-based systems, which are often:

* Time-consuming
* Prone to human error
* Inefficient for large-scale or real-time processing

With the advancement of **image processing and data acquisition technologies**, it is now possible to collect a wide range of morphological features from seeds, such as **area, perimeter, axis lengths, eccentricity, roundness, and compactness**. However, this valuable data is often underutilized in practical applications.

**The absence of an intelligent, automated classification system leads to several challenges:**

* Inconsistent or inaccurate seed type identification
* Reduced efficiency in seed packaging and distribution processes
* Increased operational costs and labor requirements
* Difficulty in maintaining uniform quality across seed batches.

**There is a need for a machine learning-based solution that can:**

* Efficiently analyze morphological seed data
* Accurately classify pumpkin seeds into predefined categories
* Support researchers, distributors, and agricultural industries in automating quality control and decision-making processes.

**Dataset Description**

The dataset used in this project contains detailed morphological and geometric measurements of various pumpkin seed varieties. These measurements are extracted from digital images of seeds and are used to classify them into specific categories. The dataset plays a crucial role in training and evaluating machine learning models for accurate seed classification.

The primary purpose of this dataset is to support the development of a predictive system that can automatically identify pumpkin seed types based on their physical characteristics.

✅ Key Features in the Dataset:

|  |  |
| --- | --- |
| **Feature Name** | **Description** |
| **Area** | Total number of pixels within the seed boundary (2D size). |
| **Perimeter** | Length of the outer boundary of the seed. |
| **Major\_Axis\_Length** | Length of the longest axis of the seed ellipse |
| **Minor\_Axis\_Length** | Length of the shortest axis of the seed ellipse. |
| **Convex\_Area** | Area of the smallest convex polygon that can contain the seed. |
| **Equiv\_Diameter** | Diameter of a circle with the same area as the seed. |
| **Eccentricity** | Ratio indicating how elongated the shape of the seed is. |
| **Solidity** | Ratio of seed area to convex area, showing how filled or compact the seed is. |
| **Extent** | Ratio of seed area to bounding box area. |
| **Roundness** | Degree to which the seed resembles a circle. |
| **Aspect\_Ratio** | Ratio of major axis to minor axis. |
| **Compactness** | Measure combining area and perimeter to reflect shape regularity. |

#### **🎯 Target Variable:**

****Class**:** This categorical feature represents the variety or type of pumpkin seed. It is the target variable for classification.

**Examples include:** **Çerçevelik, Ürgüp Sivrisi, Siyah Çekirdek, Halep, etc.**

This dataset supports multi-class classification tasks and enables the training of supervised learning algorithms to distinguish between different seed types based on quantitative image-derived features.

**2. SYSTEM ANALYSIS AND DESIGN**

This section outlines how similar systems currently function, how the proposed solution improves upon them, and what requirements (functional and non-functional) the new system must meet.

### **3.1 Existing System**

Traditional methods for classifying pumpkin seeds often rely on **manual observation** or **basic statistical analysis**. These systems:

* Do **not** use machine learning for accurate seed classification.
* Are prone to **human error** and subjective judgment.
* Require **manual feature extraction** and comparison.
* Lack interactivity or prediction capability for unseen seed data.
* Provide **no way to evaluate model performance or accuracy.**

### **3.2 Proposed System**

The proposed system uses **machine learning algorithms** to automatically classify pumpkin seed varieties based on various **morphological features** such as:

* Area
* Perimeter
* Major Axis Length
* Minor Axis Length
* Eccentricity
* Convex Area
* EquivDiameter
* Extent
* Solidity
* Roundness
* Aspect Ratio
* Compactness

The system:

* Trains on the **Pumpkin Seeds Dataset** from the UCI repository.
* Applies multiple **classification models** (Logistic Regression, KNN, SVM, Random Forest, XGBoost).
* Compares model performance using **accuracy scores**.
* Identifies the **best-performing algorithm** for prediction.
* Provides an **interactive web app** via **Streamlit** for easy use.
* Lets users input new seed measurements and get instant predictions.

**Benefits**:

* **High accuracy** in classifying seeds into correct varieties.
* **Minimizes manual labor** and human error.
* **Fast and real-time predictions** using trained models.
* Supports researchers, farmers, and agricultural labs.
* Encourages data-driven decision-making in agriculture.

### **3.3 Functional Requirements**

These are the **core functionalities** the system must provide:

* Upload pumpkin seed dataset (.xlsx or .csv format).
* Preprocess the data (label encoding, feature scaling).
* Train and test multiple classification algorithms.
* Display accuracy scores for each model.
* Allow user to enter seed measurements manually.
* Highlight the best-performing model.
* Visualize model comparisons using bar charts.
* Show predicted class based on user input.

### **3.4 Non-Functional Requirements**

These describe **qualities** the system must meet beyond functionality:

* **Usability:** Clean, friendly interface using Streamlit.
* **Performance:** App must load quickly and respond in real-time.
* **Security:** User data remains local; no server-side storage.
* **Maintainability:** Well-organized, modular code for easy updates.
* **Portability:** Should work on any system with Python and Streamlit.

**3. METHODOLOGY & IMPLEMENTATION**

**Methodology**

The methodology adopted in this project follows a structured machine learning pipeline aimed at **accurately classifying pumpkin seed varieties using morphological features**. The process involves data preparation, model training, evaluation, and deployment using an interactive dashboard.

**🔶 Step 1: Data Collection**

* The dataset is sourced from a structured **morphological analysis** of pumpkin seeds.
* It includes **shape and geometry-based** features such as Area, Perimeter, Major and Minor Axis Lengths, Eccentricity, Solidity, Compactness, and more.
* Each data point is **labeled with the corresponding seed class**, which serves as the target variable.

**🔶 Step 2: Data Preprocessing**

* The dataset is checked for **missing or inconsistent values**, and necessary cleaning is performed.
* Feature selection is applied to ensure that only **relevant columns are used for model training**.
* All numerical features are standardized using **StandardScaler to bring them into a common scale**, ensuring fair performance across different algorithms**.**

**🔶 Step 3: Data Splitting**

The data is split into:

* **Training Set (80%)**: Used to train the machine learning model.
* **Testing Set (20%)**: Used to evaluate the model’s performance.

**🔶 Step 4: Model Building**

Different regression models are applied to predict the temperature:

* Linear Regression
* Decision Tree Regressor
* Random Forest Regressor
* Support Vector Regressor (SVR)
* K-Nearest Neighbors (KNN)

**🔶 Step 5: Model Evaluation**

**Each model is evaluated using the following classification metrics:**

* Accuracy: Proportion of correctly classified instances.
* Precision: Correctness of positive predictions.
* Recall: Coverage of actual positives.
* F1-Score: Harmonic mean of precision and recall.

The model with the best overall performance across these metrics is selected for deployment.

**🔶 Step 6: Streamlit Dashboard**

* A user-friendly **Streamlit web app** is created.

It allows users to:

* Upload a CSV dataset.
* View predictions from different models.
* Input custom values and get predicted temperature.
* View model comparison chart.

**Machine Learning Models Used:**

In this project, several **regression models** were used to predict engine temperature based on engine sensor data. Each model has its own way of learning patterns from the data. Below is a short explanation of each:

**1. Linear Regression**

* It assumes a straight-line relationship between input features and the output.
* Simple and interpretable model.
* Works well when data has a linear trend.

**2. Decision Tree Regressor**

* Uses a tree-like structure to make predictions.
* Splits the data into decision nodes based on conditions.
* Easy to visualize and interpret.

**3. Random Forest Regressor**

* An ensemble of multiple decision trees.
* Combines the predictions of many trees to improve accuracy.
* Reduces overfitting compared to a single decision tree.

**4. Support Vector Regressor (SVR)**

* A powerful model that tries to fit the best line within a margin of tolerance.
* Good for high-dimensional datasets.
* Can handle nonlinear relationships using kernel functions.

**5. K-Nearest Neighbors (KNN) Regressor**

* Makes predictions based on the average output of the **k-nearest** data points.
* Simple and effective for small datasets.
* Requires proper feature scaling for better results.

Each model was evaluated using **Mean Absolute Error (MAE)** and **Root Mean Squared Error (RMSE)**. The model with the **lowest MAE** was considered the best performer for engine temperature prediction.

## **4. SYSTEM DESIGN**

System design forms the blueprint for how different modules of the pumpkin seed classification system are organized and interact. The system is built to classify pumpkin seeds into distinct varieties using multiple machine learning classification models. A simple and interactive interface is developed using **Streamlit** to facilitate easy usage, prediction, and performance visualization.

### **4.1 System Architecture**

The system architecture is modular and follows a structured pipeline comprising the following core components:

* **Data Input Layer:**Users upload a .csv or .xlsx file containing morphological features of pumpkin seeds such as Area, Perimeter, Roundness, etc.
* **Data Preprocessing Module:**This module handles missing data, encodes the target variable (Class), and standardizes all input features using StandardScaler to ensure balanced training.
* **Feature and Target Selection:**The features (like Area, Perimeter, Eccentricity, etc.) are selected for training, and the target is the seed variety (Class).
* **Model Training and Evaluation Layer:**Several machine learning models are trained including Logistic Regression, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Random Forest, and XGBoost.
* **Evaluation Metrics Generator:**After prediction, **accuracy** is calculated for each model to compare performance and select the best-performing classifier.
* **User Input and Prediction Layer:**This layer provides fields for users to enter feature values (like seed area, roundness, etc.) and get a real-time classification prediction.
* **Visualization Layer:**Uses Matplotlib and Seaborn to generate bar charts comparing model performance and to display classification results clearly.

The overall architecture ensures logical flow from data input to prediction output, enabling both automated classification and user interactivity.

### **4.2 Data Flow Diagram (DFD)**

The data flow diagram (DFD) outlines the operation sequence and how data moves across the system. The process follows this path:

1. User uploads the pumpkin seed dataset.
2. Data is cleaned, and the target column (Class) is label encoded.
3. Features and targets are separated.
4. Dataset is split into training and testing subsets.
5. Multiple classification models are trained on the training data.
6. Predictions are made on the test set.
7. Accuracy scores are calculated and compared.
8. User inputs new seed data manually.
9. The best model is used to classify the input into a seed variety.
10. Visual results (bar charts, predicted class) are displayed on the web app.

### **4.3 UML Diagrams**

UML diagrams help visualize the design and interaction between system components and the user.

#### **4.3.1 Use Case Diagram**

This diagram captures the main actions a user can perform:

* Upload pumpkin seed dataset
* Train and test multiple classification models
* Enter seed parameters for prediction
* View accuracy of each model
* Visualize model comparison and prediction output
* Each use case maps to a functional module in the system.

#### **4.3.2 Class Diagram**

The class diagram models the structure of the system using object-oriented design. Key classes include:

* DataLoader: Handles dataset upload and reading.
* Preprocessor: Encodes, cleans, and scales data.
* ModelManager: Trains various classifiers.
* Predictor: Accepts input and generates seed class predictions.
* Evaluator: Computes accuracy for model comparison.
* Visualizer: Plots charts and displays performance.

These classes work together in a pipeline for smooth data processing and prediction.

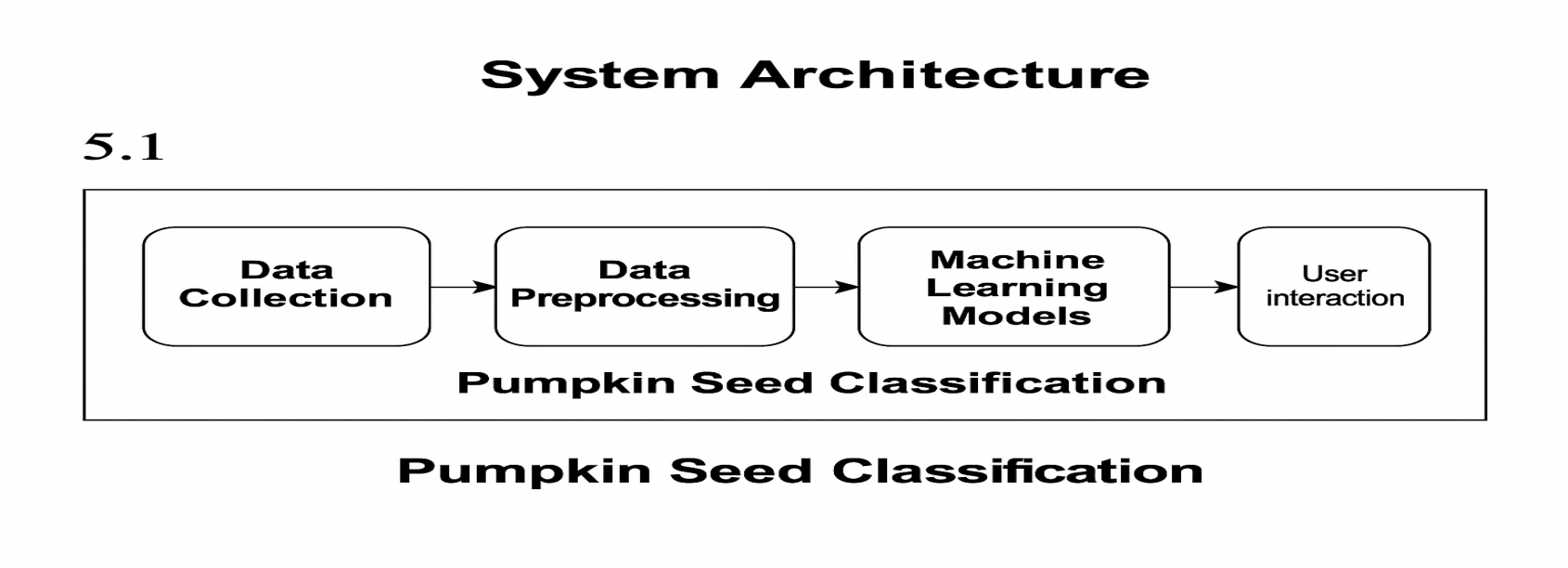
#### **4.3.3 Activity Diagram**

The activity diagram shows the dynamic flow of the system:

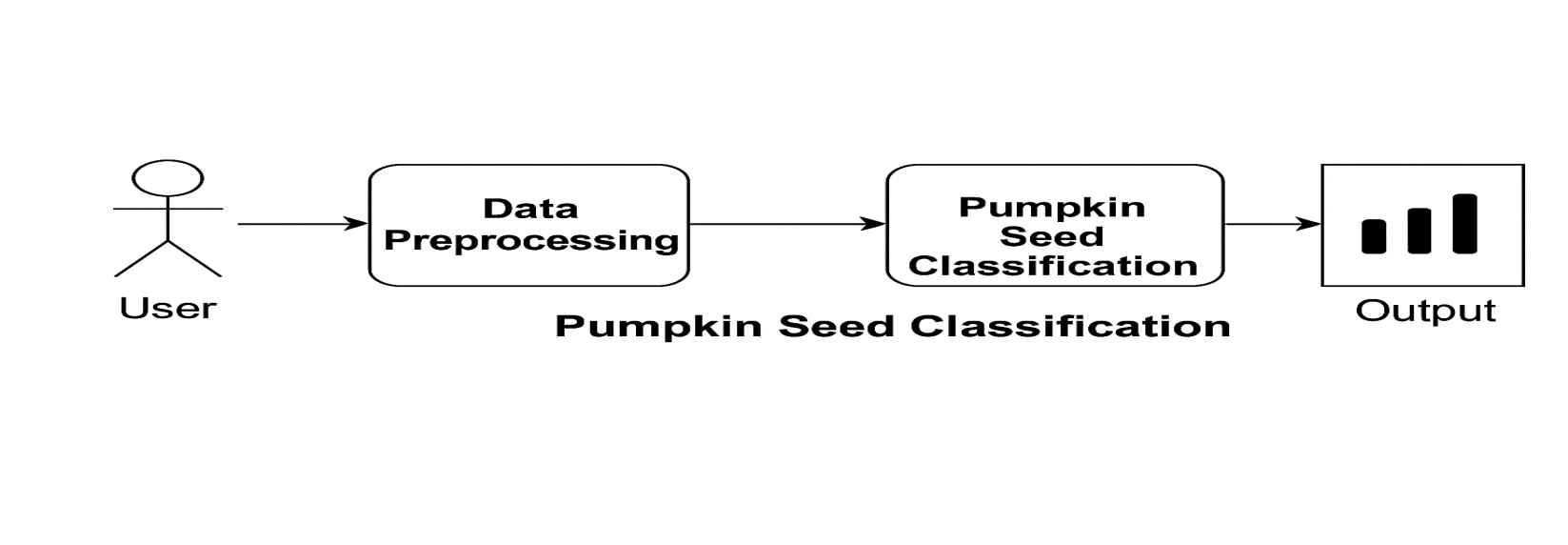
* Start: User uploads dataset
* Data cleaning and encoding
* Feature-target split
* Model training
* Accuracy evaluation
* Best model selection
* User input collection
* Classification prediction
* Visual output generation
* End

This diagram helps illustrate how control and data move through the system step-by-step.

System Design

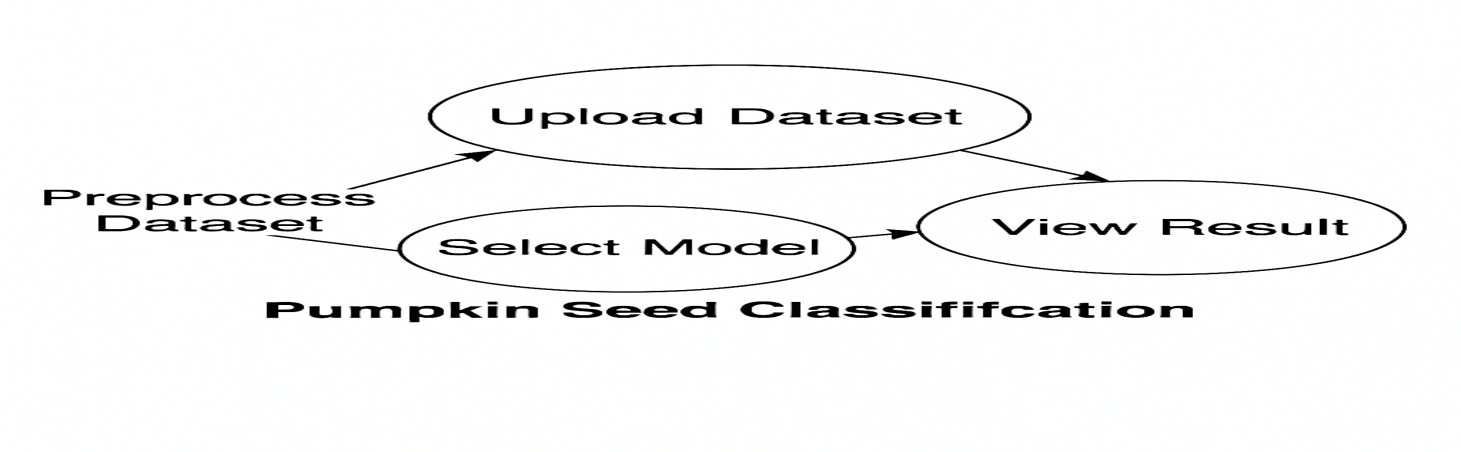


**4.2 Use Case Diagram**

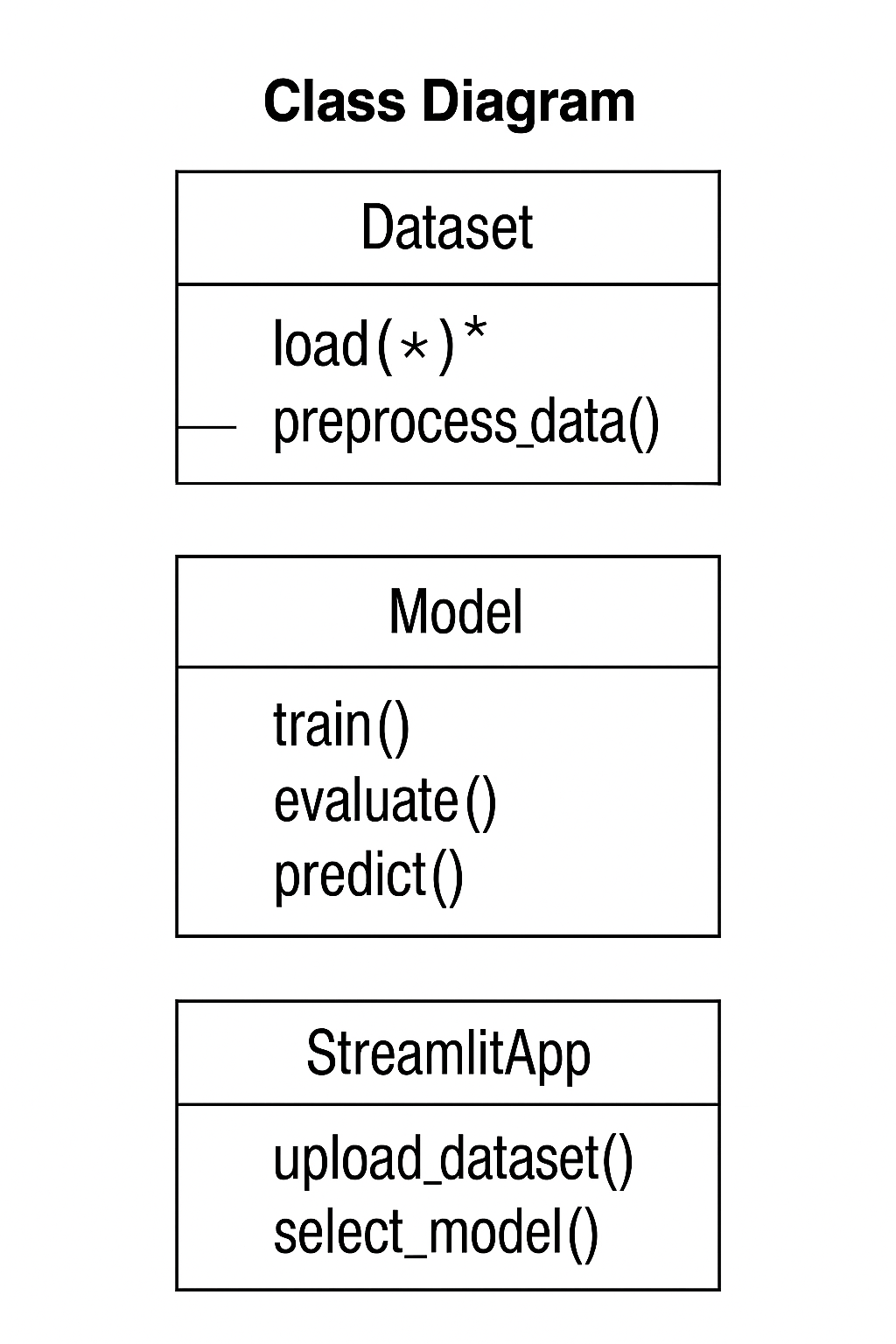


**4.3 UML Diagrams**

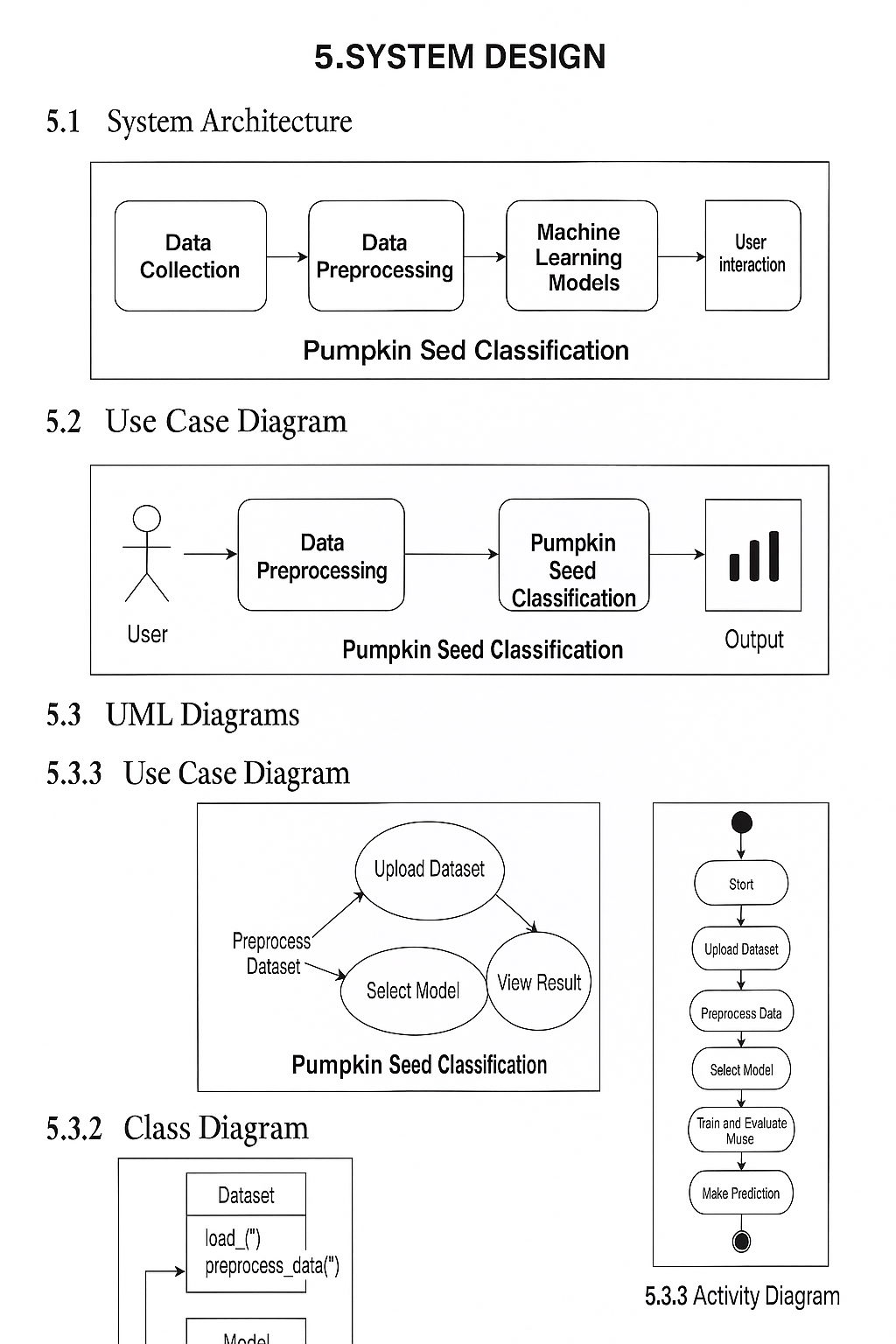
**4.3.1 Use Case Diagram**



**4.3.2. Class Diagram**



**4.3.3 Activity Diagram**



**5. IMPLEMENTATION**

**Implementation Using Streamlit**

To make the machine learning model interactive and user-friendly, we built a **web application using Streamlit**. Streamlit is a popular Python library that helps create custom web apps for machine learning and data science projects with minimal code.

* **Key Features of the Streamlit App:**

**1. File Upload Option**

* Allows users to upload the engine dataset (engine\_failure\_dataset.csv).
* Reads the uploaded file using **Pandas** and displays the preview.

**2. Data Preprocessing**

* Extracts important features like:
* RPM
* Torque
* Fuel Efficiency
* Power Output
* Vibration in X, Y, Z directions
* Scales the data using **StandardScaler** for better model performance.

**3. Model Selection**

* A dropdown menu lets the user choose from:
* Linear Regression
* Decision Tree Regressor
* Random Forest Regressor
* SVR (Support Vector Regressor)
* KNN Regressor

**4. Performance Evaluation**

* After model training, the app displays:
* **MAE (Mean Absolute Error)**
* **RMSE (Root Mean Squared Error)**

**5. Prediction on User Input**

* Users can enter custom values for all features.
* The selected model predicts the engine’s **temperature**.
* The predicted temperature is displayed along with a **bar chart**.

**6. Model Comparison**

* Compares all models based on MAE.
* Displays a comparison table and bar chart.
* Highlights the **best performing model** with the lowest error.

**Technologies Used:**

* **Python**
* **Streamlit**
* **Pandas, NumPy**
* **Matplotlib, Seaborn**
* **Scikit-learn (for models and evaluation)**

**Python:**

* A powerful and easy programming language.
* Used to write all the code in the project.
* Great for data analysis and machine learning.

**Streamlit:**

* A tool to create web apps using Python.
* Helps turn your code into a website with buttons, inputs, and outputs.
* Used to build an app for users to interact with your model.

**Pandas and NumPy:**

* **Pandas** helps read, clean, and manage data in table format (like Excel).
* **NumPy** helps with numbers and fast mathematical calculations.

**Matplotlib and Seaborn:**

* These are Python libraries used to create graphs and charts.
* **Matplotlib** makes basic plots.
* **Seaborn** makes colorful and advanced plots for better understanding.

**Scikit-learn:**

* A popular machine learning library.
* Provides ready-made algorithms like Linear Regression, Decision Trees, SVR, etc.
* Also includes tools to split data, scale features, and evaluate models.

**Results and Evaluation (MAE, RMSE):**

Once the machine learning models were trained and tested, we evaluated how well each model predicted the **engine temperature**.

To check the accuracy of the models, we used the following evaluation metrics:

**✅ Mean Absolute Error (MAE)**

* **What it means:** It tells us how far the predictions are from the actual values — on average.
* **In simple terms:** MAE shows how wrong the model is, in units of temperature (°C).
* **Lower MAE = Better Model**

**✅ Root Mean Squared Error (RMSE)**

* **What it means:** It measures the average difference between predicted and actual values, but gives **more weight to larger errors**.
* **In simple terms:** RMSE punishes big mistakes more than MAE.
* **Lower RMSE = Better Model**

**Implementation Using Streamlit**

To make the machine learning model interactive and user-friendly, we built a **web application using Streamlit**. Streamlit is a popular Python library that helps create custom web apps for machine learning and data science projects with minimal code.

* **Key Features of the Streamlit App:**

**1. File Upload Option**

* Allows users to upload the engine dataset (engine\_failure\_dataset.csv).
* Reads the uploaded file using **Pandas** and displays the preview.

**2. Data Preprocessing**

* Extracts important features like:
* RPM
* Torque
* Fuel Efficiency
* Power Output
* Vibration in X, Y, Z directions
* Scales the data using **StandardScaler** for better model performance.

**3. Model Selection**

* A dropdown menu lets the user choose from:
* Linear Regression
* Decision Tree Regressor
* Random Forest Regressor
* SVR (Support Vector Regressor)
* KNN Regressor

**4. Performance Evaluation**

* After model training, the app displays:
* **MAE (Mean Absolute Error)**
* **RMSE (Root Mean Squared Error)**

**5. Prediction on User Input**

* Users can enter custom values for all features.
* The selected model predicts the engine’s **temperature**.
* The predicted temperature is displayed along with a **bar chart**.

**6. Model Comparison**

* Compares all models based on MAE.
* Displays a comparison table and bar chart.
* Highlights the **best performing model** with the lowest error.

**Technologies Used:**

* **Python**
* **Streamlit**
* **Pandas, NumPy**
* **Matplotlib, Seaborn**
* **Scikit-learn (for models and evaluation)**

**Python:**

* A powerful and easy programming language.
* Used to write all the code in the project.
* Great for data analysis and machine learning.

**Streamlit:**

* A tool to create web apps using Python.
* Helps turn your code into a website with buttons, inputs, and outputs.
* Used to build an app for users to interact with your model.

**Pandas and NumPy:**

* **Pandas** helps read, clean, and manage data in table format (like Excel).
* **NumPy** helps with numbers and fast mathematical calculations.

**Matplotlib and Seaborn:**

* These are Python libraries used to create graphs and charts.
* **Matplotlib** makes basic plots.
* **Seaborn** makes colorful and advanced plots for better understanding.

**Scikit-learn:**

* A popular machine learning library.
* Provides ready-made algorithms like Linear Regression, Decision Trees, SVR, etc.
* Also includes tools to split data, scale features, and evaluate models.

**Results and Evaluation (MAE, RMSE):**

Once the machine learning models were trained and tested, we evaluated how well each model predicted the **engine temperature**.

To check the accuracy of the models, we used the following evaluation metrics:

**✅ Mean Absolute Error (MAE)**

* **What it means:** It tells us how far the predictions are from the actual values — on average.
* **In simple terms:** MAE shows how wrong the model is, in units of temperature (°C).
* **Lower MAE = Better Model**

**✅ Root Mean Squared Error (RMSE)**

* **What it means:** It measures the average difference between predicted and actual values, but gives **more weight to larger errors**.
* **In simple terms:** RMSE punishes big mistakes more than MAE.
* **Lower RMSE = Better Model**

**SOURCE CODE:**

**8.1 Importing Required Libraries:**

import streamlit as st

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.svm import SVR

from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error

**8.2 Page Settings and Dashboard Title:**

st.set\_page\_config(page\_title="Pumpkin Seeds ML Dashboard", layout="wide")

st.title("🎃 Pumpkin Seeds Regression Dashboard")

**8.3 Uploading and Displaying Dataset:**

uploaded\_file = st.file\_uploader("📁 Upload Pumpkin Seeds Dataset (.xlsx)", type=["xlsx"])

if uploaded\_file:

df = pd.read\_excel(uploaded\_file, engine="openpyxl")

# Data Preview

st.subheader("📄 Dataset Preview")

st.dataframe(df.head())

st.subheader("📊 Summary Statistics")

st.dataframe(df.describe())

**8.4 Feature Selection and Preprocessing:**

numeric\_cols = df.select\_dtypes(include=np.number).columns.tolist()

st.subheader("🎯 Choose Target Column")

target\_column = st.selectbox("Select the target column for regression", options=numeric\_cols)

features = [col for col in numeric\_cols if col != target\_column]

X = df[features]

y = df[target\_column]

# Data Preprocessing

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

**8.5 Model Selection and Training:**

st.subheader("🔍 Choose a Regression Model")

model\_option = st.selectbox("Select a model", ["Linear Regression", "Decision Tree", "Random Forest", "SVR", "KNN Regression"])

# Initialize model and color

model\_dict = {

"Linear Regression": (LinearRegression(), "green"),

"Decision Tree": (DecisionTreeRegressor(max\_depth=6, random\_state=0), "orange"),

"Random Forest": (RandomForestRegressor(n\_estimators=100, random\_state=0), "teal"),

"SVR": (SVR(kernel='rbf'), "purple"),

"KNN Regression": (KNeighborsRegressor(n\_neighbors=5), "blue")

}

model, color = model\_dict[model\_option]

# Train and Predict

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

# Model Evaluation

st.subheader(f"📌 {model\_option} Evaluation")

st.write("R2 Score:", round(r2\_score(y\_test, y\_pred), 4))

st.write("MAE:", round(mean\_absolute\_error(y\_test, y\_pred), 2))

st.write("RMSE:", round(np.sqrt(mean\_squared\_error(y\_test, y\_pred)), 2))

**8.6 Feature Importance for Tree-Based Models:**

if model\_option in ["Decision Tree", "Random Forest"]:

st.subheader("📊 Feature Importance")

importance = pd.Series(model.feature\_importances\_, index=features)

fig, ax = plt.subplots()

importance.sort\_values().plot(kind='barh', color=color, ax=ax)

st.pyplot(fig)

# Custom User Prediction

st.markdown("---")

st.subheader("🎯 Predict Using Custom Input")

user\_input = {feature: st.number\_input(feature, value=float(df[feature].mean()), format="%.2f") for feature in features}

user\_df = pd.DataFrame([user\_input])

user\_scaled = scaler.transform(user\_df)

prediction = model.predict(user\_scaled)[0]

st.success(f"Predicted {target\_column}: {round(prediction, 2)}")

# Prediction Visualization

st.markdown("### 📊 Prediction Graph")

fig\_pred, ax\_pred = plt.subplots(figsize=(4, 2))

ax\_pred.barh([target\_column], [prediction], color=color)

ax\_pred.set\_xlim(0, df[target\_column].max() + 10)

ax\_pred.set\_title("Predicted Output")

st.pyplot(fig\_pred)

**8.7 User Input for Prediction:**

st.markdown("---")

st.subheader("📊 Compare All Regression Models")

if st.button("Compare Models"):

models = {

"Linear Regression": LinearRegression(),

"Decision Tree": DecisionTreeRegressor(max\_depth=6, random\_state=0),

"Random Forest": RandomForestRegressor(n\_estimators=100, random\_state=0),

"SVR": SVR(kernel='rbf'),

"KNN Regression": KNeighborsRegressor(n\_neighbors=5)

}

results = []

for name, m in models.items():

m.fit(X\_train, y\_train)

preds = m.predict(X\_test)

results.append([

name,

r2\_score(y\_test, preds),

mean\_absolute\_error(y\_test, preds),

np.sqrt(mean\_squared\_error(y\_test, preds))

])

results\_df = pd.DataFrame(results, columns=["Model", "R2 Score", "MAE", "RMSE"])

best\_model = results\_df.sort\_values(by="R2 Score", ascending=False).iloc[0]["Model"]

st.dataframe(results\_df)

st.markdown("### 📈 R2 Score Comparison")

fig\_cmp, ax\_cmp = plt.subplots()

sns.barplot(data=results\_df, x="R2 Score", y="Model", palette="coolwarm", ax=ax\_cmp)

st.pyplot(fig\_cmp)

st.success(f"🏆 Best Performing Model: {best\_model}")

else:

st.info("📅 Please upload your Pumpkin\_Seeds\_Dataset.xlsx to begin.")

**COMPLETE SOURCE CODE :**

import streamlit as st

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.svm import SVR

from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error

# Page Configuration

st.set\_page\_config(page\_title="Pumpkin Seeds ML Dashboard", layout="wide")

st.title("🎃 Pumpkin Seeds Regression Dashboard")

# File Upload

uploaded\_file = st.file\_uploader("📁 Upload Pumpkin Seeds Dataset (.xlsx)", type=["xlsx"])

if uploaded\_file:

df = pd.read\_excel(uploaded\_file, engine="openpyxl")

# Data Preview

st.subheader("📄 Dataset Preview")

st.dataframe(df.head())

st.subheader("📊 Summary Statistics")

st.dataframe(df.describe())

# Feature Selection

numeric\_cols = df.select\_dtypes(include=np.number).columns.tolist()

st.subheader("🎯 Choose Target Column")

target\_column = st.selectbox("Select the target column for regression", options=numeric\_cols)

features = [col for col in numeric\_cols if col != target\_column]

X = df[features]

y = df[target\_column]

# Data Preprocessing

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

# Model Selection

st.subheader("🔍 Choose a Regression Model")

model\_option = st.selectbox("Select a model", ["Linear Regression", "Decision Tree", "Random Forest", "SVR", "KNN Regression"])

# Initialize model and color

model\_dict = {

"Linear Regression": (LinearRegression(), "green"),

"Decision Tree": (DecisionTreeRegressor(max\_depth=6, random\_state=0), "orange"),

"Random Forest": (RandomForestRegressor(n\_estimators=100, random\_state=0), "teal"),

"SVR": (SVR(kernel='rbf'), "purple"),

"KNN Regression": (KNeighborsRegressor(n\_neighbors=5), "blue")

}

model, color = model\_dict[model\_option]

# Train and Predict

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

# Model Evaluation

st.subheader(f"📌 {model\_option} Evaluation")

st.write("R2 Score:", round(r2\_score(y\_test, y\_pred), 4))

st.write("MAE:", round(mean\_absolute\_error(y\_test, y\_pred), 2))

st.write("RMSE:", round(np.sqrt(mean\_squared\_error(y\_test, y\_pred)), 2))

# Feature Importance

if model\_option in ["Decision Tree", "Random Forest"]:

st.subheader("📊 Feature Importance")

importance = pd.Series(model.feature\_importances\_, index=features)

fig, ax = plt.subplots()

importance.sort\_values().plot(kind='barh', color=color, ax=ax)

st.pyplot(fig)

# Custom User Prediction

st.markdown("---")

st.subheader("🎯 Predict Using Custom Input")

user\_input = {feature: st.number\_input(feature, value=float(df[feature].mean()), format="%.2f") for feature in features}

user\_df = pd.DataFrame([user\_input])

user\_scaled = scaler.transform(user\_df)

prediction = model.predict(user\_scaled)[0]

st.success(f"Predicted {target\_column}: {round(prediction, 2)}")

# Prediction Visualization

st.markdown("### 📊 Prediction Graph")

fig\_pred, ax\_pred = plt.subplots(figsize=(4, 2))

ax\_pred.barh([target\_column], [prediction], color=color)

ax\_pred.set\_xlim(0, df[target\_column].max() + 10)

ax\_pred.set\_title("Predicted Output")

st.pyplot(fig\_pred)

# Model Comparison

st.markdown("---")

st.subheader("📊 Compare All Regression Models")

if st.button("Compare Models"):

models = {

"Linear Regression": LinearRegression(),

"Decision Tree": DecisionTreeRegressor(max\_depth=6, random\_state=0),

"Random Forest": RandomForestRegressor(n\_estimators=100, random\_state=0),

"SVR": SVR(kernel='rbf'),

"KNN Regression": KNeighborsRegressor(n\_neighbors=5)

}

results = []

for name, m in models.items():

m.fit(X\_train, y\_train)

preds = m.predict(X\_test)

results.append([

name,

r2\_score(y\_test, preds),

mean\_absolute\_error(y\_test, preds),

np.sqrt(mean\_squared\_error(y\_test, preds))

])

results\_df = pd.DataFrame(results, columns=["Model", "R2 Score", "MAE", "RMSE"])

best\_model = results\_df.sort\_values(by="R2 Score", ascending=False).iloc[0]["Model"]

st.dataframe(results\_df)

st.markdown("### 📈 R2 Score Comparison")

fig\_cmp, ax\_cmp = plt.subplots()

sns.barplot(data=results\_df, x="R2 Score", y="Model", palette="coolwarm", ax=ax\_cmp)

st.pyplot(fig\_cmp)

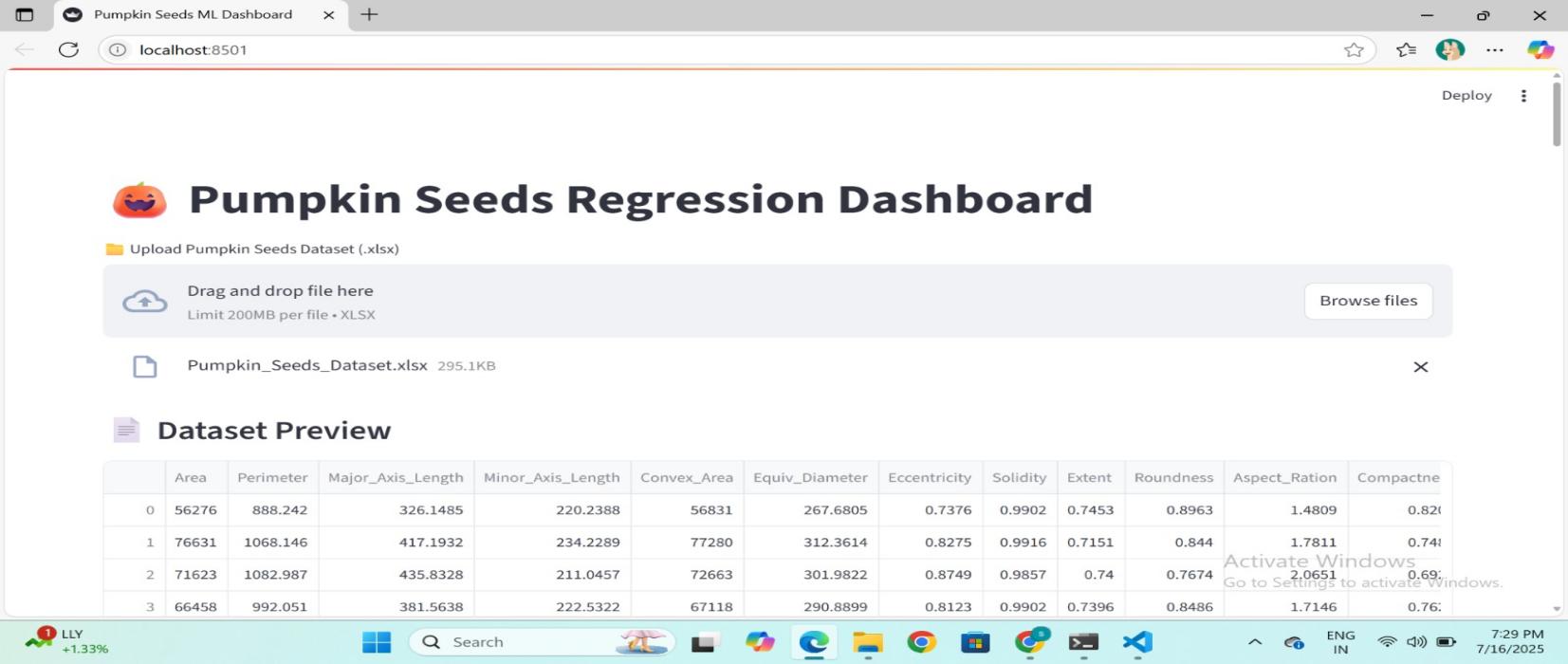
st.success(f"🏆 Best Performing Model: {best\_model}")

else:

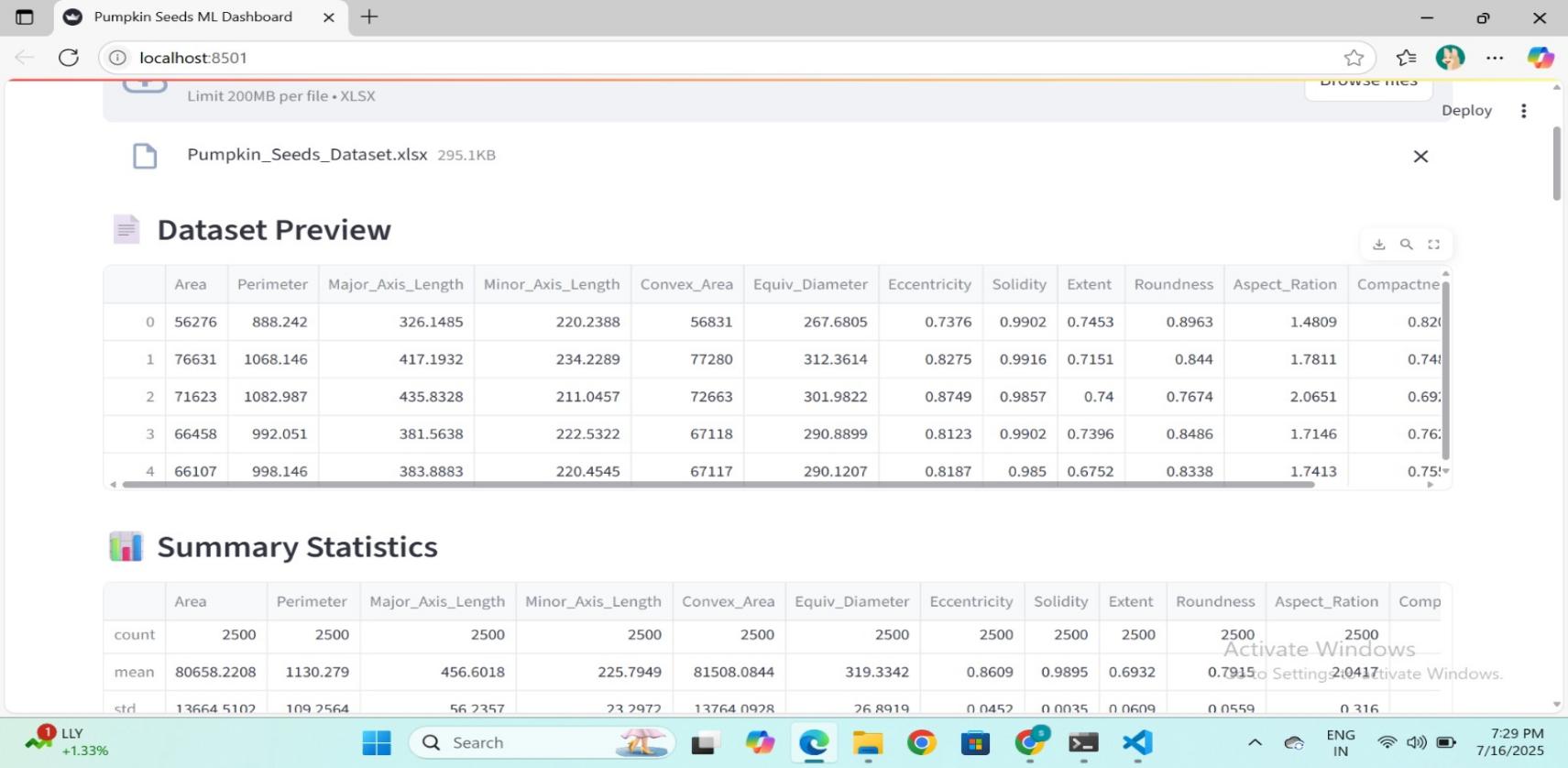
st.info("📅 Please upload your Pumpkin\_Seeds\_Dataset.xlsx to begin.")

**Result:**

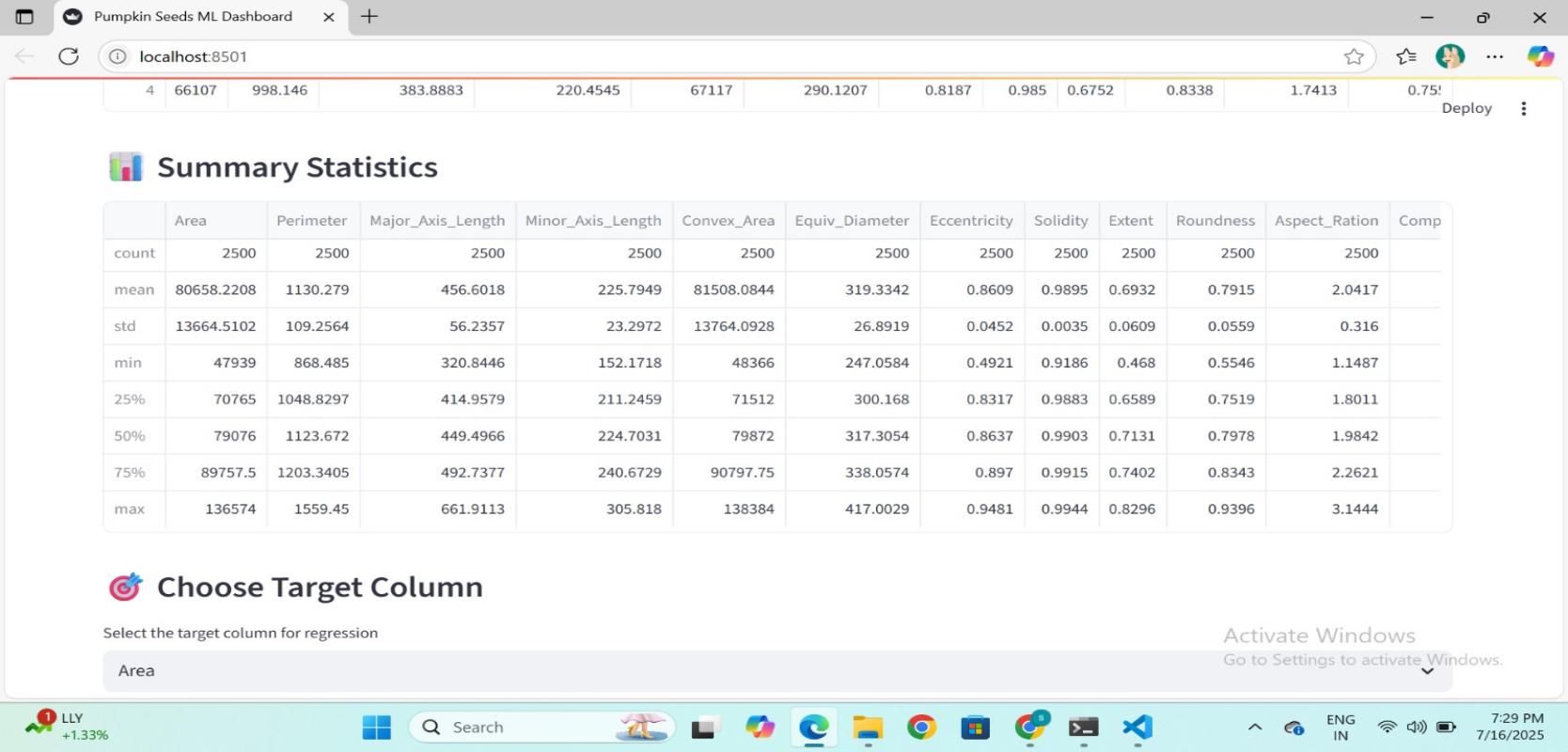
**Output 1:**



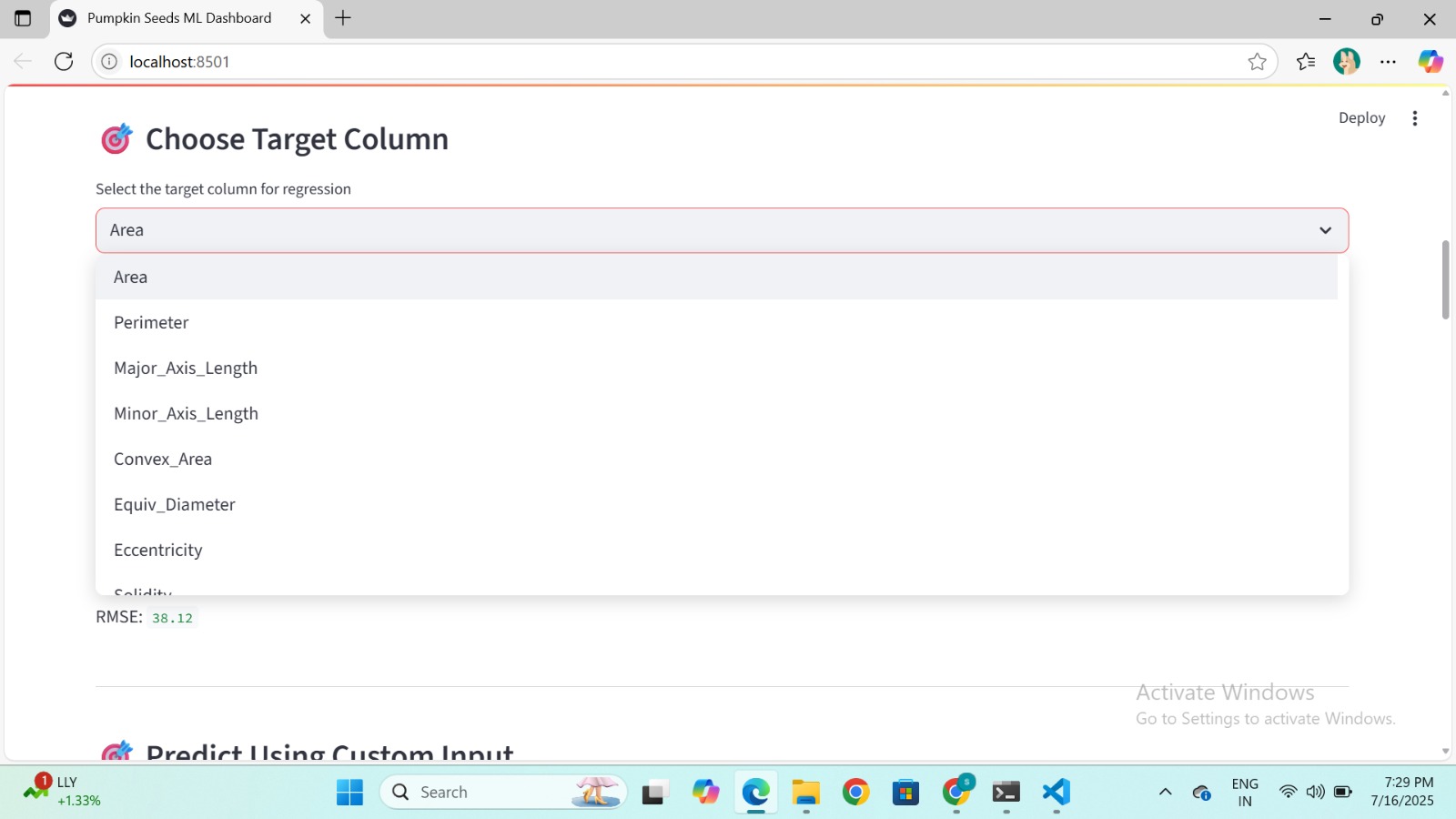
**Output 2:**



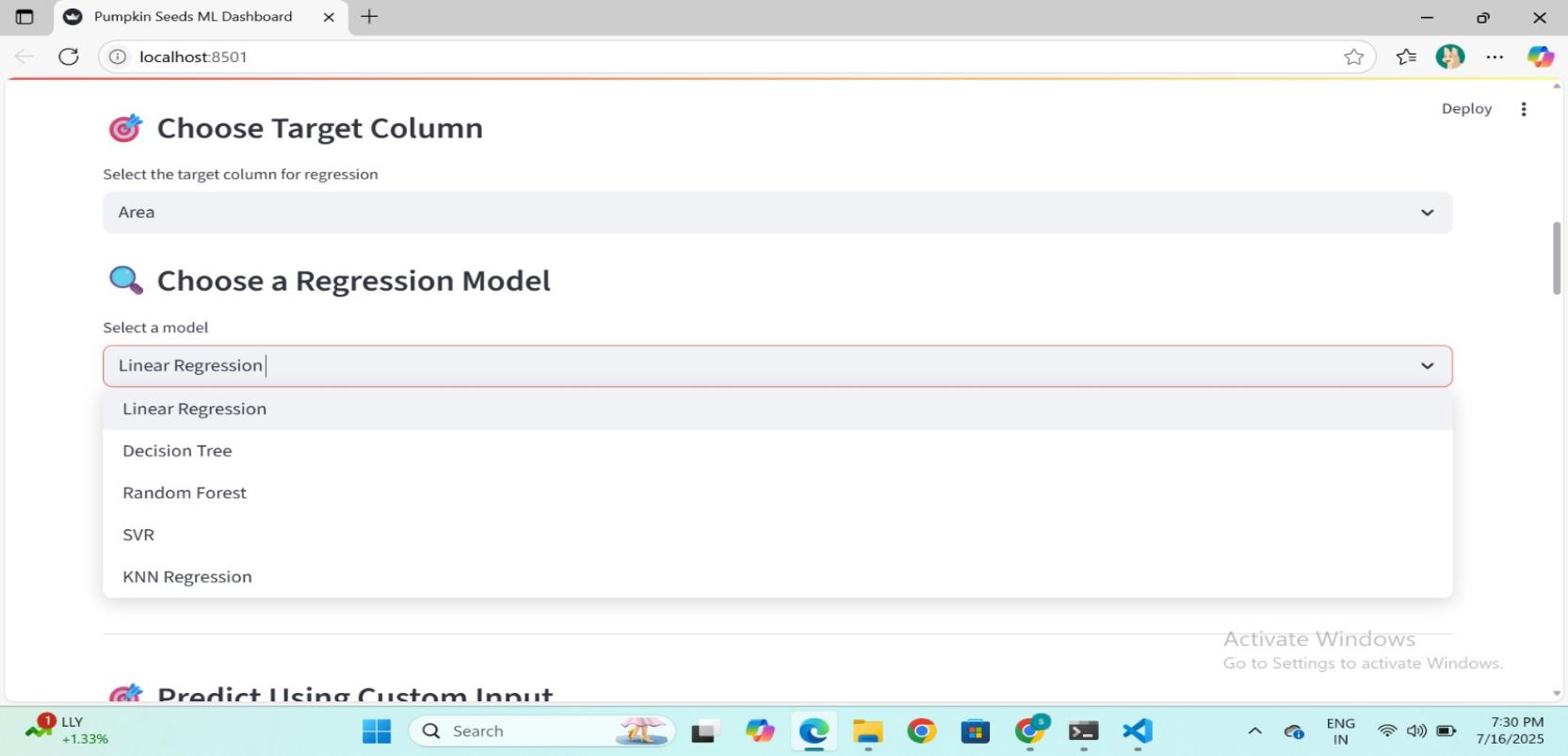
**Output 3:**



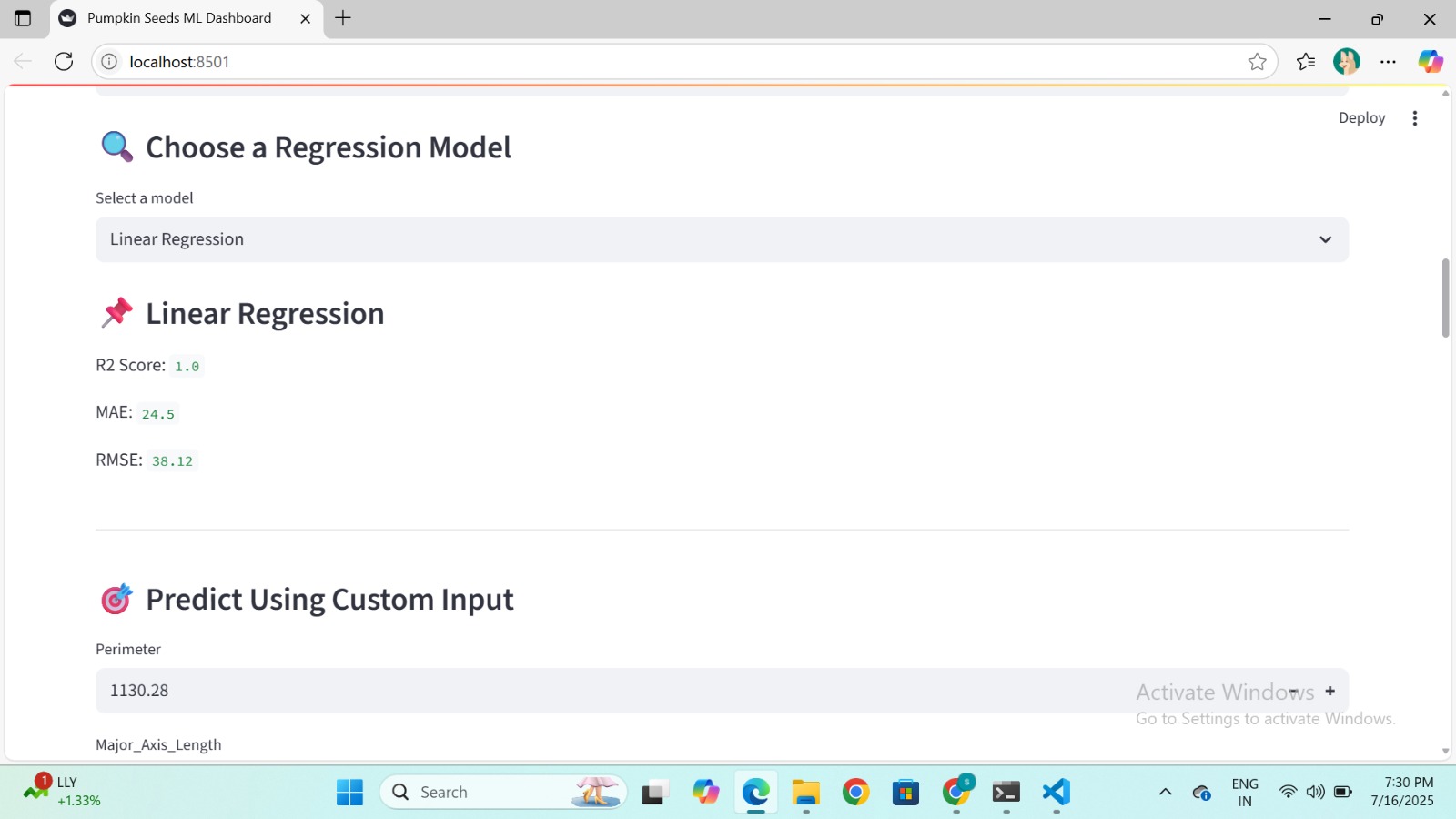
**Output 4:**



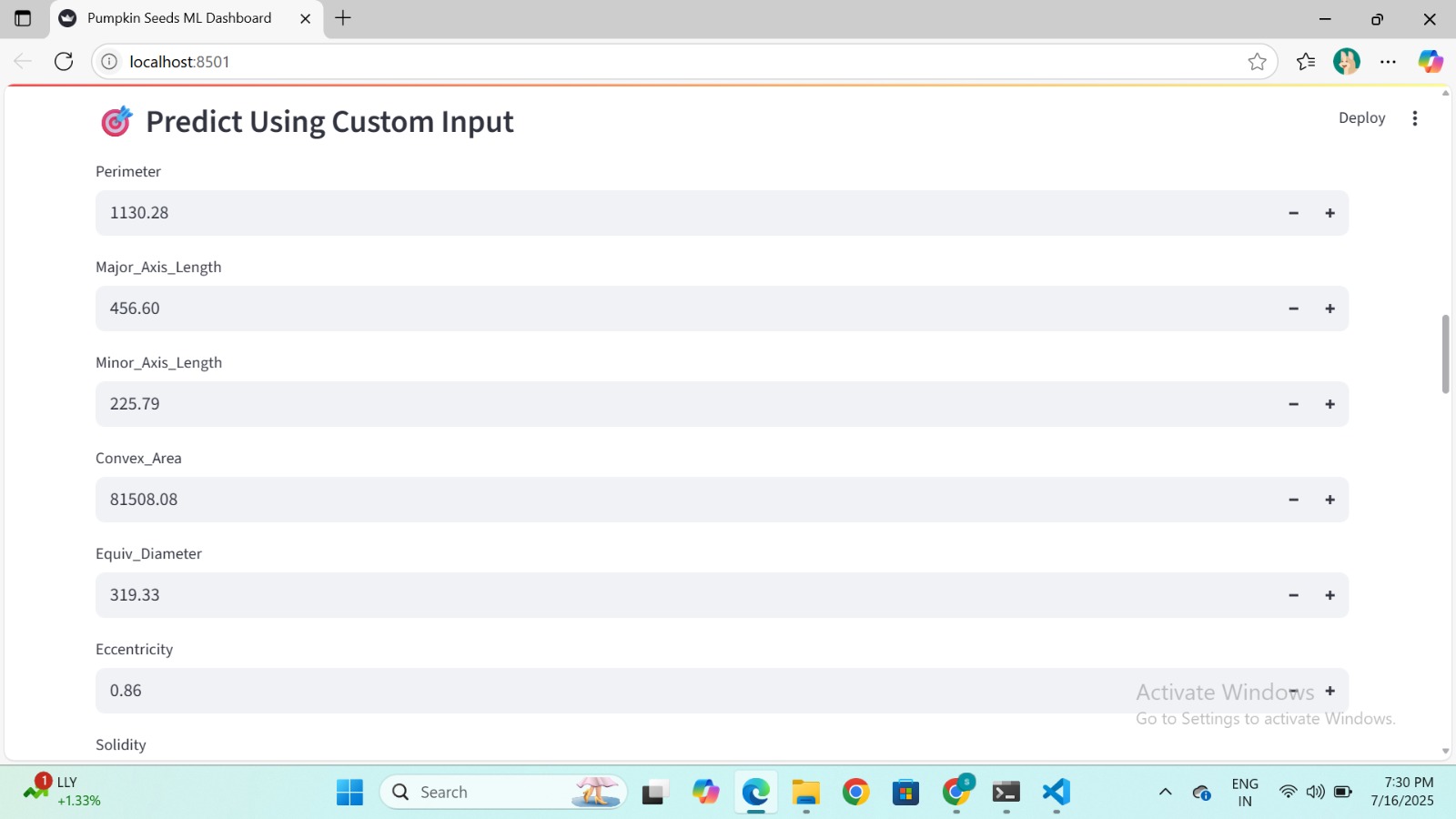
**Output 5:**



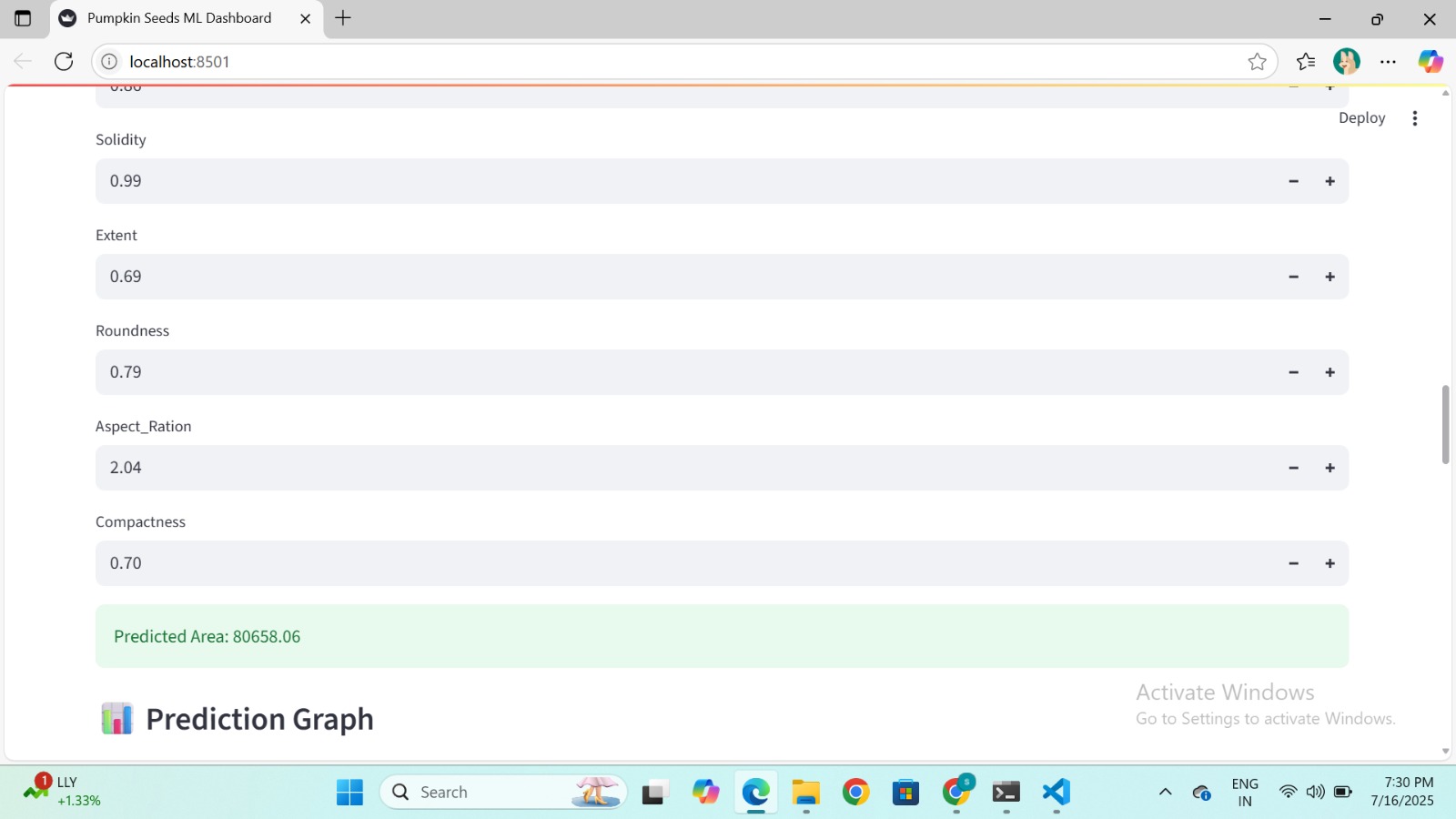
**Output 6:**



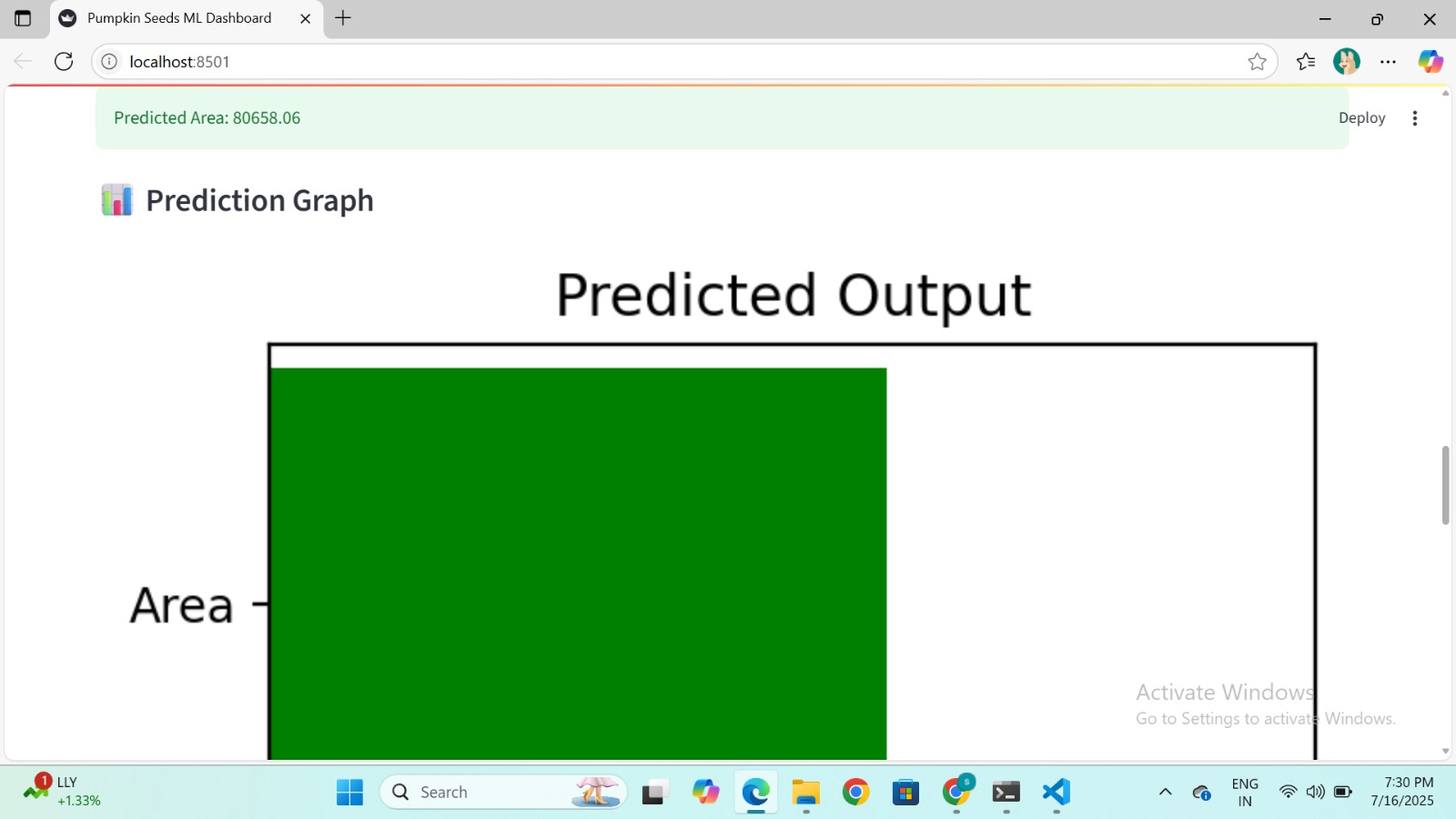
**Output 7:**



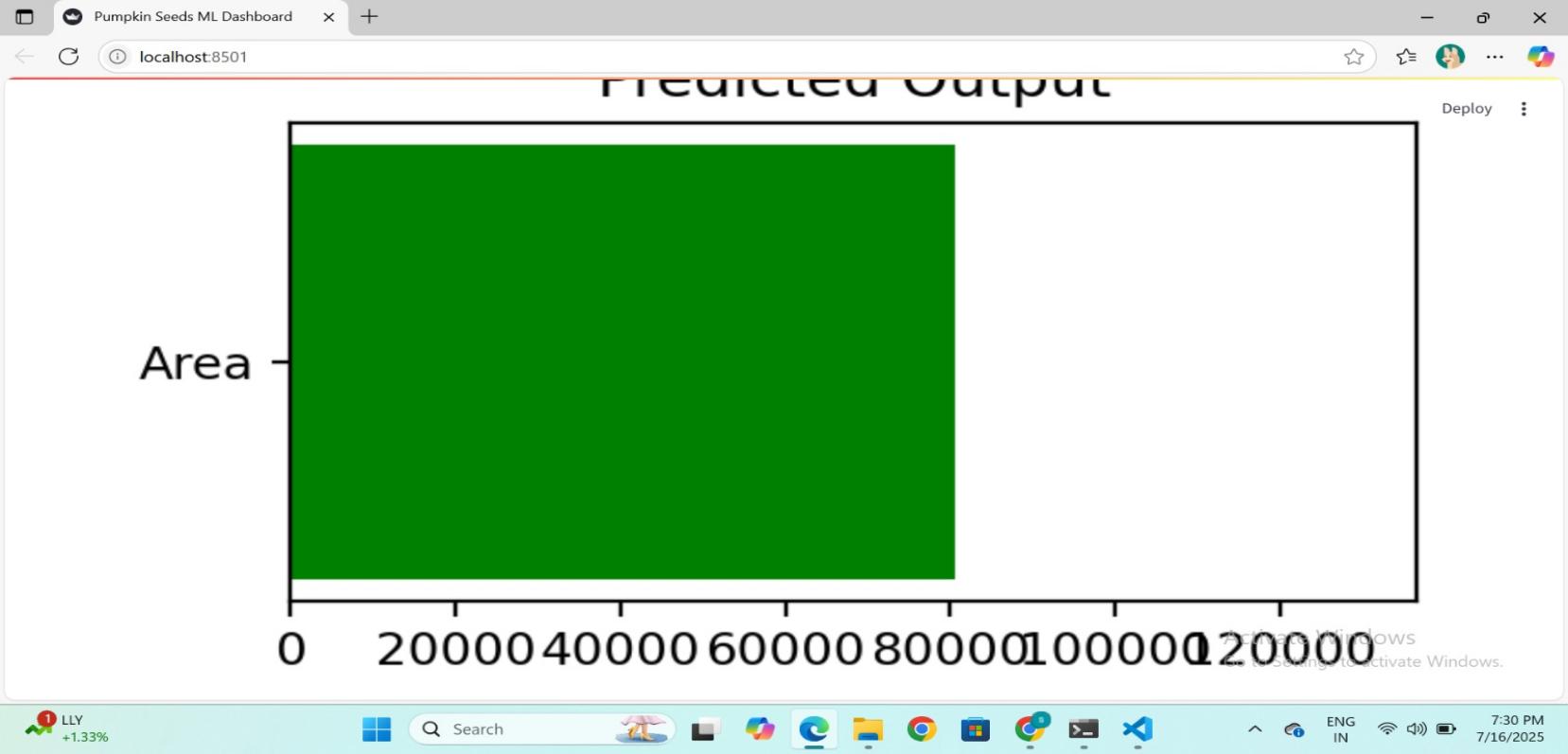
**Output 8:**



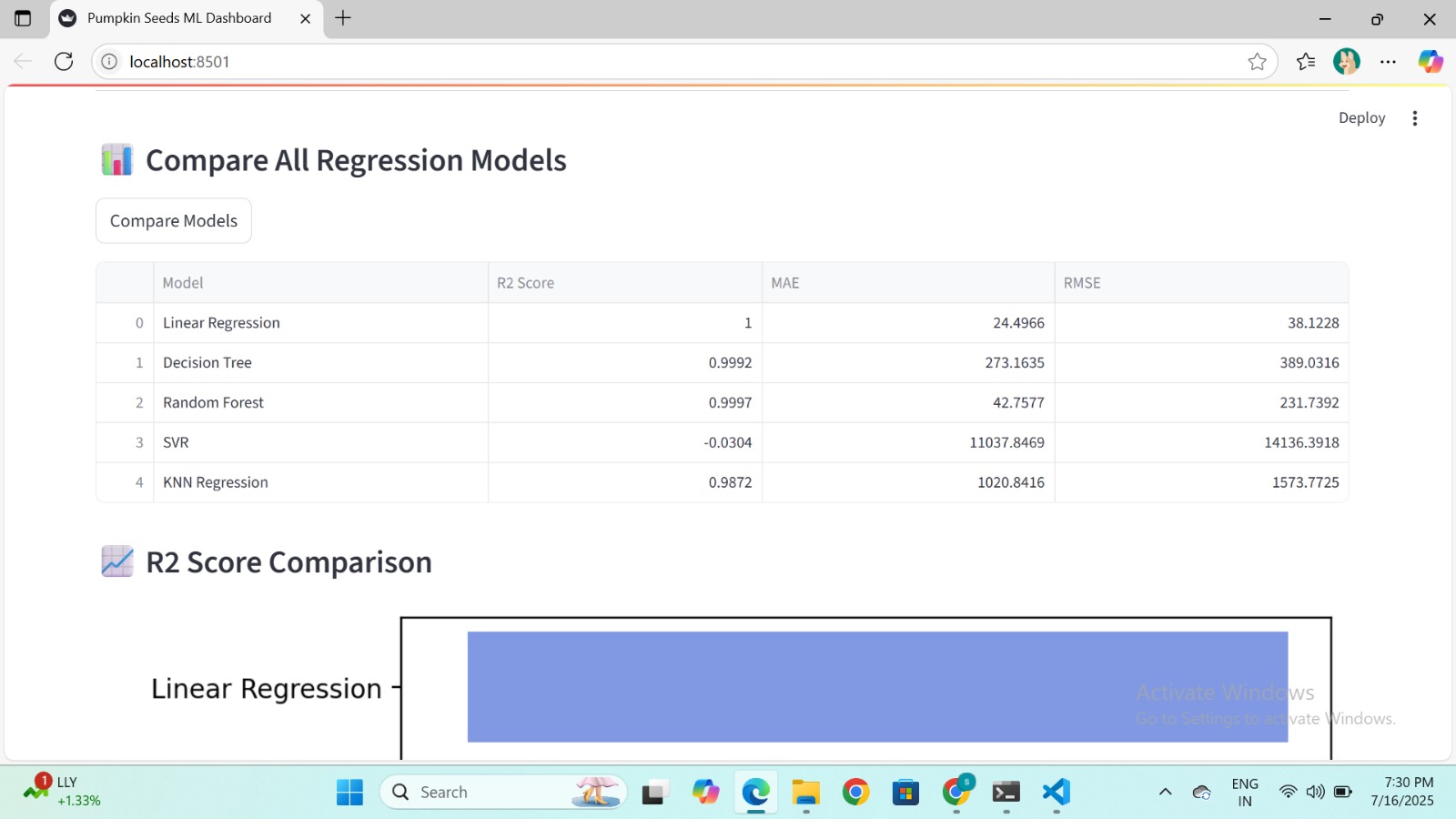
**Output 9:**



**Output 10:**



**Output 11:**



**Output 12:**



Output 13:



### **6. TESTING**

Testing plays a vital role in ensuring the correctness, reliability, and performance of the Pumpkin Seeds regression system. Different levels of testing were conducted to verify that all components—from data handling to model predictions—work as expected.

### **6.1 Unit Testing**

Unit testing was performed on individual components to confirm that each part functions correctly in isolation. In the pumpkin seeds project, unit tests were applied to:

* Data loading and preprocessing functions
* Feature selection and scaling logic
* Regression model implementations (e.g., Linear Regression, Random Forest)

### **6.2 Integration Testing**

Integration testing ensured that the interaction between various modules (data → model → prediction) worked seamlessly:

* Confirmed that StandardScaler correctly transformed user and training data
* Verified predictions were generated without runtime errors
* Ensured user input was properly captured, transformed, and passed into the model

This level of testing validated the flow from dataset upload to live predictions inside the Streamlit app.

### **6.3 System Testing**

System testing was conducted on the entire Streamlit application to evaluate end-to-end functionality:

* The web interface loads and performs all required actions
* File upload, model training, prediction, and graph generation work smoothly
* Regression metrics like R2, MAE, and RMSE are consistent with offline evaluation
* Validated correct behavior for edge-case inputs and error handling

Through this testing process, we ensured that the system delivers a robust, user-friendly experience for analyzing and predicting values in the Pumpkin Seeds dataset.

**Conclusion**

The project “Pumpkin Seed Classification Using Machine Learning” successfully demonstrates how modern machine learning techniques can be applied to real-world agricultural challenges, specifically in automating the classification of seed varieties based on morphological data.

By analyzing features such as **Area, Perimeter, Major and Minor Axis Lengths, Eccentricity, Solidity, Roundness, Aspect Ratio, and Compactness**, the system implemented and compared several supervised classification algorithms, including:

* Logistic Regression
* Decision Tree Classifier
* Random Forest Classifier
* Support Vector Machine (SVM)
* K-Nearest Neighbors (KNN)

The best-performing model was selected based on evaluation metrics such as **Accuracy, Precision, Recall,** and **F1-Score**, ensuring high reliability and consistency in predicting the correct seed class.

To enhance accessibility and practical usability, the final model was integrated into an interactive **Streamlit web application**, which allows users to:

* Upload custom seed data,
* Visualize classification predictions,
* Compare model performance through real-time metrics and charts.

Key outcomes of this project:

* Automated, fast, and accurate classification of pumpkin seed types.
* Reduced dependency on manual inspection and potential human error.
* Scalable and adaptable system for similar tasks in agriculture and quality control.

This project provides a robust and intelligent solution to streamline seed classification and quality assurance in agricultural industries, thereby contributing to higher productivity and standardization using data-driven methods.

**References**

1. Scikit-learn: Machine Learning in Python – [https://scikit-learn.org/](https://scikit-learn.org/" \t "_new)
2. Streamlit: The fastest way to build data apps – [https://streamlit.io/](https://streamlit.io/" \t "_new)
3. NumPy Documentation – [https://numpy.org/doc/](https://numpy.org/doc/" \t "_new)
4. Pandas Documentation – [https://pandas.pydata.org/](https://pandas.pydata.org/" \t "_new)
5. Matplotlib Documentation – [https://matplotlib.org/stable/index.html](https://matplotlib.org/stable/index.html" \t "_new)
6. Seaborn Documentation – [https://seaborn.pydata.org/](https://seaborn.pydata.org/" \t "_new)
7. Dataset: Pumpkin Seeds Dataset (UCI Machine Learning Repository) – Provided for academic use
8. Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media.
9. Raschka, S., & Mirjalili, V. (2019). Python Machine Learning. Packt Publishing.